Course Enrollment System

Team 3

Yiqing Huang

Liang Chen

Jiangting Wan

Ekta Pant

Yuke Tu

Table of Contents

Core Features

- **Student Management** Enables student registration, login, profile course browsing and course registration.
- **Course Management** Maintains course info, enforces capacity limits, classifies by credit level.
- **Instructor Management** Allows instructors to view courses and manage student grades.

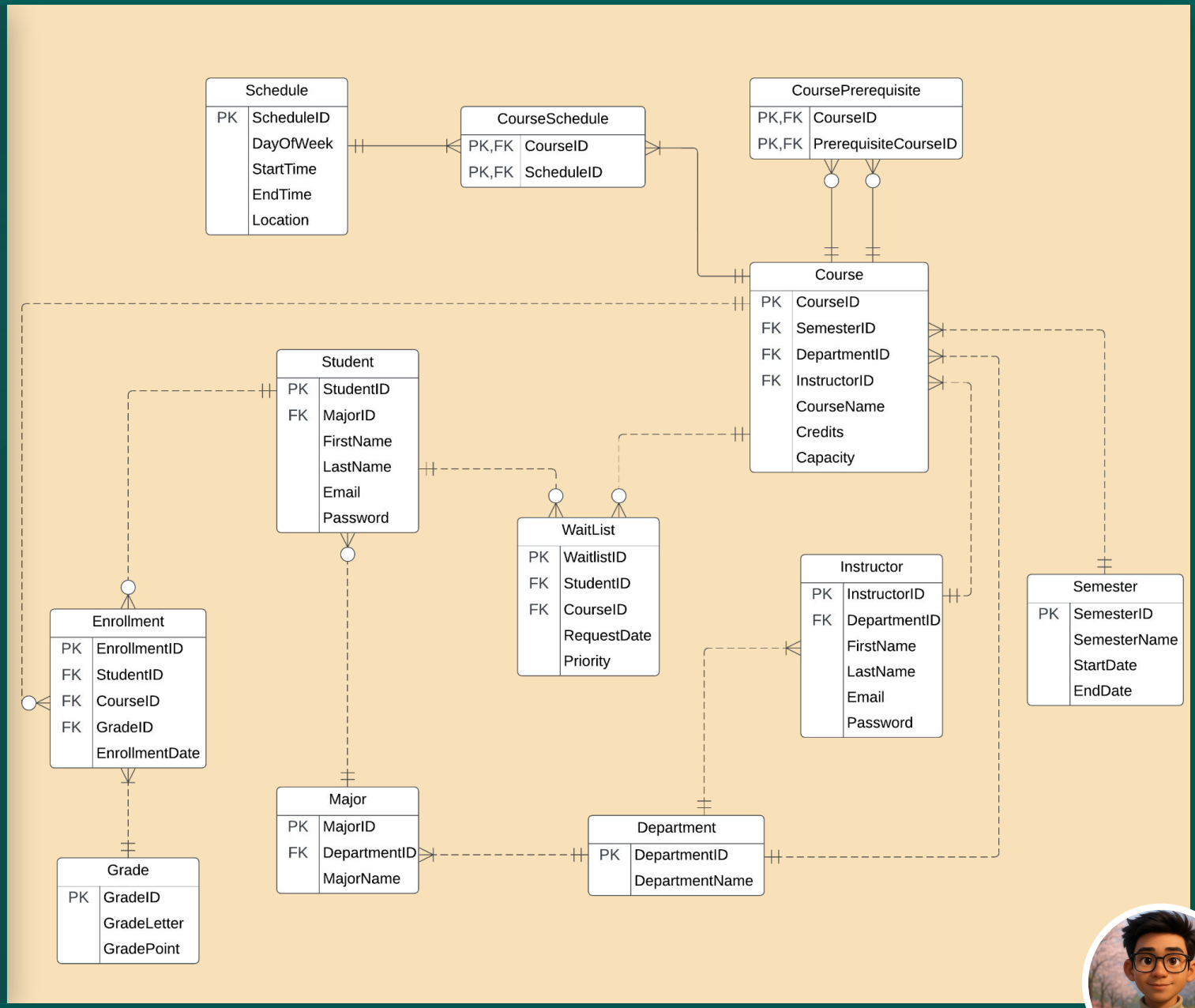
- **Enrollment Process** Supports course selection by semester,credit level or subject area. Automatically adds students to waitlist when courses are full.
- **Visualization** Provides views into students' basic information, enrollment details, and academic performance.

Business Problem Addressed

- Implemented full course capacity control and waitlist mechanism with automatic queuing and priority handling to ensure fair enrollment.
- Enabled multi-slot scheduling by modeling many-to-many relationships between courses and time slots, supporting complex timetable requirements.
- Secured student and instructor accounts using symmetric encryption with certificates and a master key, enhancing overall system security.
- Built a fully connected data model linking courses, instructors, semesters, schedules, and enrollments to maintain consistency across entities.

Database Architecture

Entity-Relationship Diagram



Core Entities

- **Student** - Personal information and academic records
- **Course** - Class details, capacity, and credit information
- **Instructor** - Faculty details and departmental affiliations
- **Enrollment** - Student course registrations and grades
- **Waitlist** - Queue management for full courses

Table Code Implement

Enrollment:

```
CREATE TABLE Course
(
  CourseID INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
  SemesterID INT NOT NULL,
  DepartmentID INT NOT NULL,
  InstructorID INT NOT NULL,
  CourseName VARCHAR(100) NOT NULL,
  Credits INT NOT NULL,
  Capacity INT NOT NULL,
  FOREIGN KEY (SemesterID)  REFERENCES Semester(SemesterID),
  FOREIGN KEY (DepartmentID) REFERENCES
Department(DepartmentID),
  FOREIGN KEY (InstructorID)  REFERENCES Instructor(InstructorID)
);
```

Table Code Implement

Student:

```
CREATE TABLE Student
(
  StudentID INT IDENTITY(1,1) NOT
NULL PRIMARY KEY,
  MajorID INT NOT NULL,
  FirstName VARCHAR(50) NOT NULL,
  LastName VARCHAR(50) NOT NULL,
  Email VARCHAR(50) NOT NULL,
  Password VARBINARY(256) NOT
NULL,
  FOREIGN KEY (MajorID) REFERENCES
Major(MajorID)
);
```

Course:

```
CREATE TABLE Course
(
  CourseID INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
  SemesterID INT NOT NULL,
  DepartmentID INT NOT NULL,
  InstructorID INT NOT NULL,
  CourseName VARCHAR(100) NOT NULL,
  Credits INT NOT NULL,
  Capacity INT NOT NULL,
  FOREIGN KEY (SemesterID)  REFERENCES Semester(SemesterID),
  FOREIGN KEY (DepartmentID) REFERENCES
Department(DepartmentID),
  FOREIGN KEY (InstructorID)  REFERENCES Instructor(InstructorID)
);
```

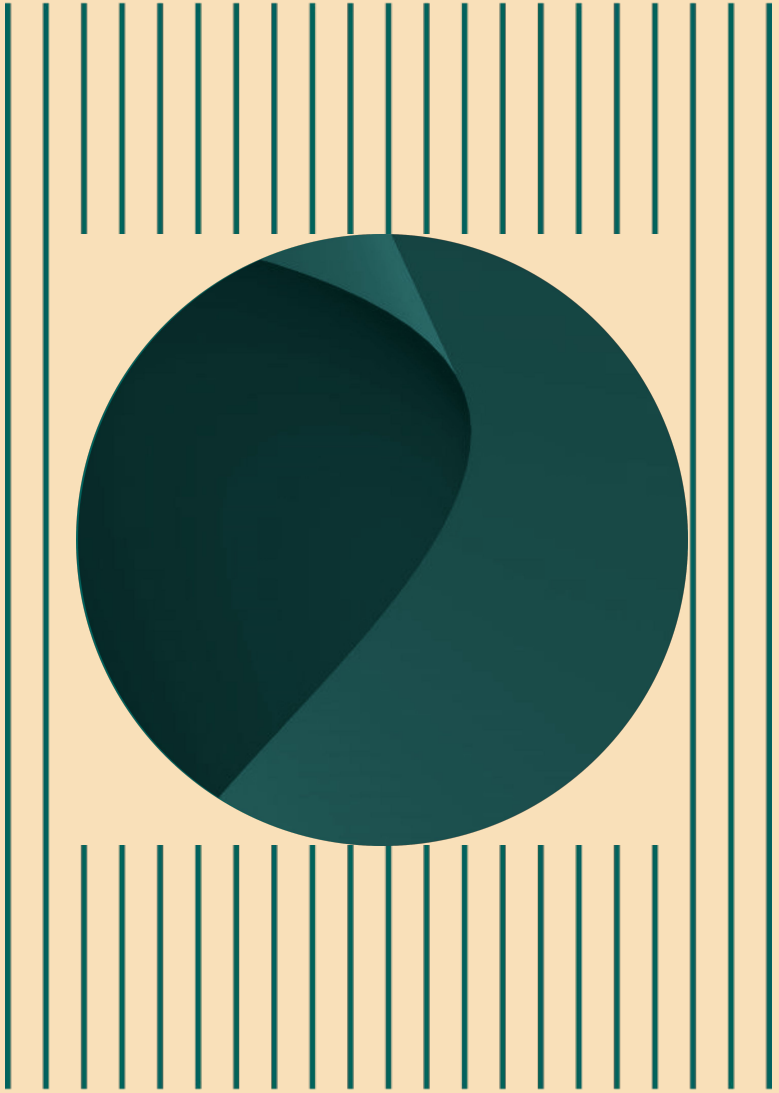
Table Code Implement

Instructor:

```
CREATE TABLE Student
(
  StudentID INT IDENTITY(1,1) NOT
NULL PRIMARY KEY,
  MajorID INT NOT NULL,
  FirstName VARCHAR(50) NOT NULL,
  LastName VARCHAR(50) NOT NULL,
  Email VARCHAR(50) NOT NULL,
  Password VARBINARY(256) NOT
NULL,
  FOREIGN KEY (MajorID) REFERENCES
Major(MajorID)
);
```

Waitlist:

```
CREATE TABLE Waitlist
(
  WaitlistID INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
  StudentID INT NOT NULL,
  CourseID INT NOT NULL,
  RequestDate DATE NOT NULL DEFAULT GETDATE(),
  Priority INT NOT NULL,
  FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
  FOREIGN KEY (CourseID)  REFERENCES Course(CourseID)
);
```



Database Design Decisions

3NF Implementation

Minimizes data redundancy

Associative Entities

Handles many-to-many relationships

Password Encryption

Secures user credentials

Waitlist Priority System

Ensures fair enrollment process

Core Functions and Views

Stored Procedure: sp_RegisterCourse

1. **Pre-check: Enrollment & Waitlist Status**
   Verify if the student is already enrolled or waitlisted.
2. **Course Validation & Capacity Check**
   Confirm the course exists and check available seats.
3. **Enroll Student if Space Available**
   If seats are open, register the student successfully.
4. **Waitlist Handling if Course Full**
   Add to waitlist if available, or return failure if both full.

```
CREATE PROCEDURE sp_RegisterCourse
   @StudentID INT,
   @CourseID INT,
   @ResultMessage VARCHAR(200) OUTPUT
AS
BEGIN
   SET NOCOUNT ON;
   BEGIN TRAN;
   DECLARE @Capacity INT,
           @EnrolledCount INT,
           @WaitlistCount INT,
           @WaitlistLimit INT = 15;  -- set a limit for the waitlist

   -- check if the student is already enrolled in the course
   IF EXISTS ( SELECT 1 FROM Enrollment
               WHERE StudentID = @StudentID AND CourseID = @CourseID)
   BEGIN
       SET @ResultMessage = 'Already enrolled in the course.' ;
       ROLLBACK TRANSACTION;
       RETURN;
   END

   -- check if the student is already on the waitlist
   IF EXISTS ( SELECT 1 FROM Waitlist
               WHERE StudentID = @StudentID AND CourseID = @CourseID)
   BEGIN
       SET @ResultMessage = 'Already on the waitlist for this course.' ;
       ROLLBACK TRANSACTION;
       RETURN;
   END

   -- check if the course exists
   SELECT @Capacity = Capacity
   FROM Course
   WHERE CourseID = @CourseID;

   IF @Capacity IS NULL
   BEGIN
        SET @ResultMessage = 'Course not found' ;
        ROLLBACK TRANSACTION;
        RETURN;
   END

SELECT @EnrolledCount = COUNT(*)
   FROM Enrollment
   WHERE CourseID = @CourseID;

   -- check if the course is full, if not, enroll the student
   IF @EnrolledCount < @Capacity
   BEGIN
        INSERT INTO Enrollment (StudentID, CourseID, EnrollmentDate)
        VALUES (@StudentID, @CourseID, GETDATE());
        SET @ResultMessage = 'Successfully enrolled in the course.' ;
        COMMIT TRANSACTION;
        RETURN;
   END
   ELSE
   BEGIN

        SELECT @WaitlistCount = COUNT(*)
        FROM Waitlist
        WHERE CourseID = @CourseID;

        -- check if the waitlist is full, if not,
          add the student to the waitlist, otherwise, rollback
        IF @WaitlistCount < @WaitlistLimit
        BEGIN
             INSERT INTO Waitlist (StudentID, CourseID, RequestDate, Priority)
             VALUES (@StudentID, @CourseID, GETDATE(), @WaitlistCount + 1);
             SET @ResultMessage = 'Course is full; you have been added to the waitlist.' ;
             COMMIT TRANSACTION;
             RETURN;
        END
        ELSE
        BEGIN
             SET @ResultMessage = 'Course and waitlist are both full. Registration
failed.';
             ROLLBACK TRANSACTION;
             RETURN;
        END
   END
END;
```

Computed Columns

Format Full Name: Combines First Name and Last Name

```
CREATE FUNCTION dbo.fn_FormatFullName
(
  @FirstName VARCHAR(50),
  @LastName  VARCHAR(50)
)
RETURNS VARCHAR(101)
WITH SCHEMABINDING
AS
BEGIN
  RETURN (@FirstName + ' ' + @LastName);
END;

ALTER TABLE Student
ADD FullName AS dbo.fn_FormatFullName(FirstName, LastName) PERSISTED;

ALTER TABLE Instructor
ADD FullName AS dbo.fn_FormatFullName(FirstName, LastName) PERSISTED;
```

CourseLevel: Categorizes course as Basic, Intermediate, Advance

```
CREATE FUNCTION dbo.fn_CourseLevel
(
  @Credits INT
)
RETURNS VARCHAR(50)
WITH SCHEMABINDING
AS
BEGIN
  DECLARE @Level VARCHAR(50);
  IF @Credits >= 4
     SET @Level = 'Advanced';
  ELSE IF @Credits = 3
     SET @Level = 'Intermediate';
  ELSE
     SET @Level = 'Basic';
  RETURN @Level;
END;

ALTER TABLE Course
ADD CourseLevel AS dbo.fn_CourseLevel(Credits) PERSISTED;
```

Check Constraints

Check Capacity: Ensures course capacity is between 5 and 100

```
CREATE FUNCTION dbo.fn_CheckCapacity(@Capacity INT)
RETURNS BIT
AS
BEGIN
  IF @Capacity BETWEEN 5 AND 100
     RETURN 1;
  RETURN 0;
END;

ALTER TABLE Course
ADD CONSTRAINT CK_Course_Capacity CHECK
(dbo.fn_CheckCapacity(Capacity) = 1);
```

Check Grade Point: Ensures grade point is between 0.00 and 4.00

```
CREATE FUNCTION dbo.fn_CheckGradePoint(@GradePoint
DECIMAL(3,2))
RETURNS BIT
AS
BEGIN
  IF @GradePoint BETWEEN 0 AND 4.0
     RETURN 1;
  RETURN 0;
END;

ALTER TABLE Grade
ADD CONSTRAINT CK_Grade_GradePoint CHECK
(dbo.fn_CheckGradePoint(GradePoint) = 1);
```

Views

- **Student Basic Info** - Displays student names, emails, majors, and departments.
- **Department Course Summary**: Shows each department's course count and course list.
- **Course Schedule Details**: Lists course schedules by day, time, instructor, and semester.
- **Course Registration Status**: Tracks course capacity, enrollment count, remaining seats, and registration status.
- **Enrollment Details**: Combines student, course, instructor, and grade information for enrollment history.

	StudentID	FullName	Email	MajorName	DepartmentName
1	1	Alice Smith	alice@example.com	Software Engineering	Computer Science
2	2	Bob Johnson	bob@example.com	Data Science	Computer Science
3	3	Carol Williams	carol@example.com	Applied Mathematics	Mathematics
4	4	David Brown	david@example.com	Astrophysics	Physics
5	5	Eve Jones	eve@example.com	Organic Chemistry	Chemistry
6	6	Frank Garcia	frank@example.com	Microbiology	Biology
7	7	Grace Miller	grace@example.com	Ancient History	History
8	8	Hank Davis	hank@example.com	Literature	English
9	9	Ivy Rodriguez	ivy@example.com	Finance	Economics
10	10	Jack Martinez	jack@example.com	Philosophy	Philosophy

	DepartmentID	DepartmentName	CourseCount	CourseList
1	1	Computer Science	20	Intro to Programming, Data Structures, Algorithms, Computer Organization,
2	2	Mathematics	2	Calculus I, Calculus II
3	3	Physics	1	Physics I
4	4	Chemistry	1	Organic Chemistry
5	5	Biology	1	Molecular Biology
6	6	History	1	World History
7	7	English	1	English Literature
8	8	Economics	1	Microeconomics
9	9	Philosophy	1	Introduction to Philosophy
10	10	Psychology	1	Cognitive Psychology

	EnrollmentID	StudentID	StudentName	CourseName	SemesterName	InstructorName	EnrollmentDate	GradeLetter
1	1	1	Alice Smith	Intro to Programming	Fall 2022	Alice Wang	2024-04-05	A
2	2	2	Bob Johnson	Intro to Programming	Fall 2022	Alice Wang	2024-04-06	A-
3	3	3	Carol Williams	Data Structures	Spring 2023	Bob Chen	2024-04-07	B+
4	4	4	David Brown	Data Structures	Spring 2023	Bob Chen	2024-04-08	B
5	5	5	Eve Jones	Algorithms	Summer 2023	Carol Li	2024-04-09	A
6	6	6	Frank Garcia	Algorithms	Summer 2023	Carol Li	2024-04-10	A-
7	7	7	Grace Miller	Computer Organization	Fall 2023	David Zhang	2024-04-11	B+
8	8	8	Hank Davis	Operating Systems	Spring 2024	Alice Wang	2024-04-12	B
9	9	9	Ivy Rodriguez	Databases	Summer 2024	David Zhang	2024-04-13	A
10	10	10	Jack Martinez	Networks	Fall 2024	Carol Li	2024-04-14	A-
11	11	1	Alice Smith	Software Engineering	Spring 2025	Carol Li	2024-12-05	[NULL]
12	12	2	Bob Johnson	Software Engineering	Spring 2025	Carol Li	2024-12-06	[NULL]
13	13	3	Carol Williams	Software Engineering	Spring 2025	Carol Li	2024-12-07	[NULL]
14	14	4	David Brown	Software Engineering	Spring 2025	Carol Li	2024-12-08	[NULL]
15	15	5	Eve Jones	Software Engineering	Spring 2025	Carol Li	2024-12-09	[NULL]
16	16	6	Frank Garcia	Software Engineering	Spring 2025	Carol Li	2024-12-10	[NULL]
17	17	7	Grace Miller	Software Engineering	Spring 2025	Carol Li	2024-12-11	[NULL]
18	18	1	Alice Smith	Artificial Intelligence	Summer 2025	Bob Chen	2025-02-10	[NULL]
19	19	2	Bob Johnson	Artificial Intelligence	Summer 2025	Bob Chen	2025-02-11	[NULL]
20	20	3	Carol Williams	Artificial Intelligence	Summer 2025	Bob Chen	2025-02-12	[NULL]

```sql
-- Create views for the Student Information
CREATE VIEW vw_StudentBasicInfo
AS
SELECT
  s.StudentID,
  s.FullName,
  s.Email,
  m.MajorName,
  d.DepartmentName
FROM Student s
JOIN Major m ON s.MajorID = m.MajorID
JOIN Department d ON m.DepartmentID = d.DepartmentID;

-- Create views for the Each Department's Courses
CREATE VIEW vw_DepartmentCourseSummary
AS
SELECT
  d.DepartmentID,
  d.DepartmentName,
  COUNT(c.CourseID) AS CourseCount,
  STUFF(
      (SELECT ', ' + c2.CourseName
       FROM Course c2
       WHERE c2.DepartmentID = d.DepartmentID
       FOR XML PATH(''), TYPE
      ).value('.', 'NVARCHAR(MAX)'), 1, 2, ''
  ) AS CourseList
FROM Department d
LEFT JOIN Course c ON d.DepartmentID = c.DepartmentID
GROUP BY d.DepartmentID, d.DepartmentName;

-- Create views for the Student Enrollment details
CREATE VIEW vw_EnrollmentDetails
AS
  SELECT
      e.EnrollmentID,
      s.StudentID,
      s.FullName AS StudentName,
      c.CourseName,
      sem.SemesterName,
      i.FullName AS InstructorName,
      e.EnrollmentDate,
      g.GradeLetter
  FROM Enrollment e
      JOIN Student s ON e.StudentID = s.StudentID
      JOIN Course c ON e.CourseID = c.CourseID
      JOIN Semester sem ON c.SemesterID = sem.SemesterID
      JOIN Instructor i ON c.InstructorID = i.InstructorID
      LEFT JOIN Grade g ON e.GradeID = g.GradeID;
```

```sql
-- Create views for the Course Schedule details
CREATE VIEW vw_CourseScheduleDetails
AS
  SELECT
     c.CourseID,
     c.CourseName,
     sem.SemesterName,
     i.FullName AS InstructorName,
     s.DayOfWeek,
     CONVERT(VARCHAR(5), s.StartTime, 108) AS StartTime,
     CONVERT(VARCHAR(5), s.EndTime, 108) AS EndTime
  FROM Course c
     JOIN Semester sem ON c.SemesterID = sem.SemesterID
     JOIN Instructor i ON c.InstructorID = i.InstructorID
     JOIN CourseSchedule cs ON c.CourseID = cs.CourseID
     JOIN Schedule s ON cs.ScheduleID = s.ScheduleID;
```

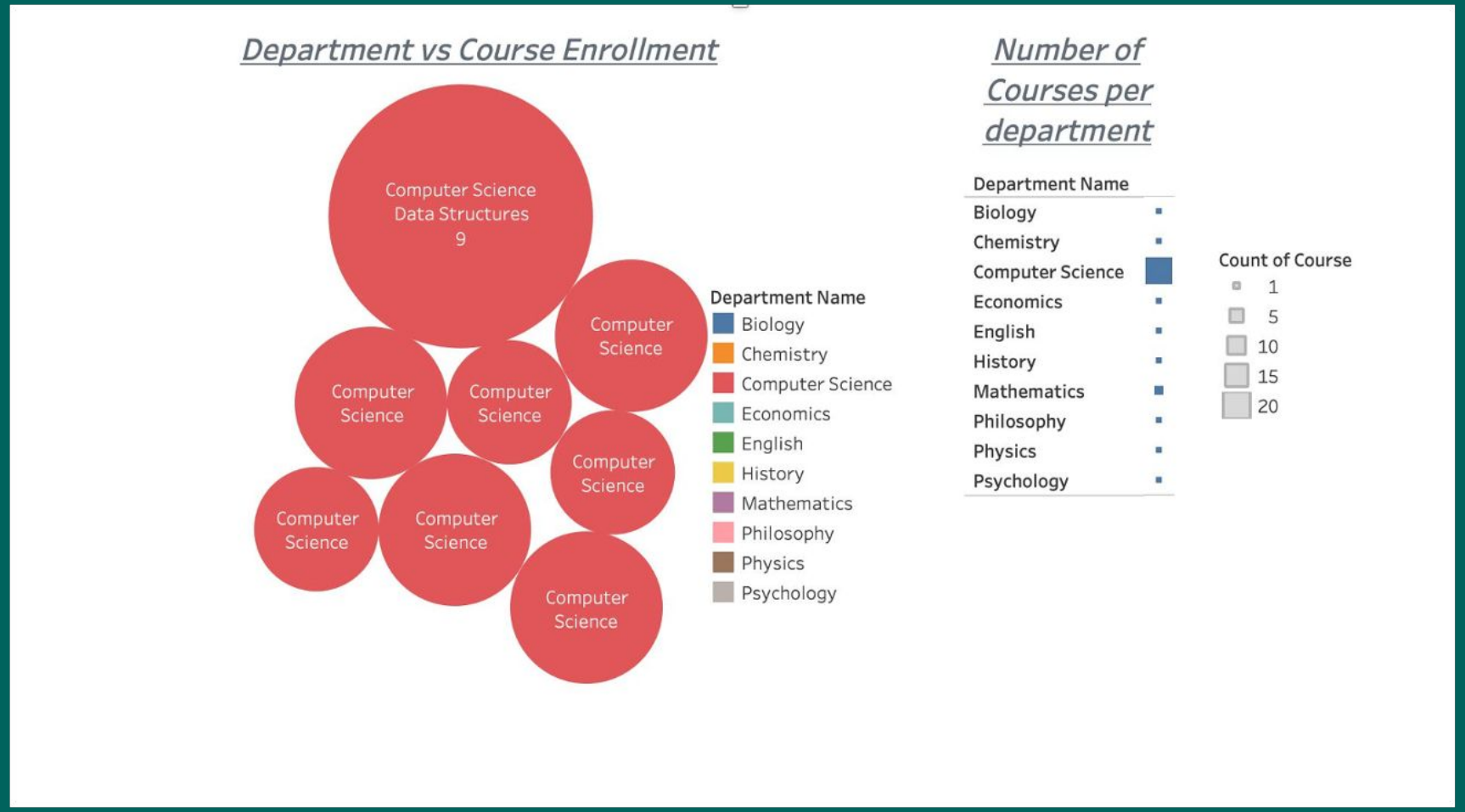
	CourseID	CourseName	SemesterName	InstructorName	DayOfWeek	StartTime	EndTime
1	1	Intro to Programming	Fall 2022	Alice Wang	Monday	09:00	12:00
2	2	Data Structures	Spring 2023	Bob Chen	Tuesday	10:00	13:00
3	3	Algorithms	Summer 2023	Carol Li	Wednesday	09:00	10:30
4	4	Computer Organization	Fall 2023	David Zhang	Thursday	10:00	11:30
5	5	Operating Systems	Spring 2024	Alice Wang	Friday	12:00	15:00
6	6	Databases	Summer 2024	David Zhang	Monday	13:00	14:30
7	7	Networks	Fall 2024	Carol Li	Tuesday	13:00	15:00
8	8	Software Engineering	Spring 2025	Carol Li	Wednesday	18:30	21:30
9	9	Artificial Intelligence	Summer 2025	Bob Chen	Thursday	15:00	16:30
10	10	Machine Learning	Fall 2025	David Zhang	Friday	15:00	18:00
11	11	Computer Graphics	Fall 2022	Alice Wang	Monday	09:00	12:00
12	12	Web Programming	Spring 2023	Carol Li	Tuesday	10:00	13:00
13	13	Mobile Application Development	Summer 2023	Alice Wang	Wednesday	09:00	10:30
14	14	Cybersecurity Fundamentals	Fall 2023	David Zhang	Thursday	10:00	11:30
15	15	Cloud Computing	Spring 2024	Bob Chen	Friday	12:00	15:00
16	16	Parallel Computing	Summer 2024	Carol Li	Monday	13:00	14:30
17	17	Programming Languages	Fall 2024	Alice Wang	Tuesday	13:00	15:00
18	18	Compiler Construction	Spring 2025	David Zhang	Wednesday	18:30	21:30
19	19	Distributed Systems	Summer 2025	Carol Li	Thursday	15:00	16:30
20	20	Big Data Analytics	Fall 2025	Bob Chen	Friday	15:00	18:00

	CourseID	CourseName	SemesterName	Capacity	EnrolledCount	RemainingSeats	RegistrationStatus
1	1	Intro to Programming	Fall 2022	40	2	38	Closed
2	2	Data Structures	Spring 2023	35	2	33	Closed
3	3	Algorithms	Summer 2023	30	2	28	Closed
4	4	Computer Organization	Fall 2023	40	1	39	Closed
5	5	Operating Systems	Spring 2024	30	1	29	Closed
6	6	Databases	Summer 2024	40	1	39	Closed
7	7	Networks	Fall 2024	30	1	29	Closed
8	8	Software Engineering	Spring 2025	40	7	33	Open
9	9	Artificial Intelligence	Summer 2025	35	5	30	Open
10	10	Machine Learning	Fall 2025	30	5	25	Open
11	11	Computer Graphics	Fall 2022	40	0	40	Closed
12	12	Web Programming	Spring 2023	35	0	35	Closed
13	13	Mobile Application Development	Summer 2023	30	0	30	Closed
14	14	Cybersecurity Fundamentals	Fall 2023	40	0	40	Closed
15	15	Cloud Computing	Spring 2024	30	0	30	Closed
16	16	Parallel Computing	Summer 2024	40	0	40	Closed
17	17	Programming Languages	Fall 2024	35	0	35	Closed
18	18	Compiler Construction	Spring 2025	30	0	30	Open
19	19	Distributed Systems	Summer 2025	40	0	40	Open
20	20	Big Data Analytics	Fall 2025	30	0	30	Open

```sql
-- Create views for the Course Registration status
CREATE VIEW vw_CourseRegistrationStatus
AS
SELECT
   c.CourseID,
   c.CourseName,
   sem.SemesterName,
   c.Capacity,
   ISNULL(e.EnrolledCount, 0) AS EnrolledCount,
   (c.Capacity - ISNULL(e.EnrolledCount, 0)) AS RemainingSeats,
   CASE
      WHEN sem.EndDate < GETDATE() THEN 'Closed'
      WHEN ISNULL(e.EnrolledCount, 0) >= c.Capacity THEN 'Full'
      WHEN ISNULL(e.EnrolledCount, 0) = 0 THEN 'Open'
      ELSE 'Open'
   END AS RegistrationStatus
FROM Course c
JOIN Semester sem ON c.SemesterID = sem.SemesterID
LEFT JOIN (
   SELECT CourseID, COUNT(*) AS EnrolledCount
   FROM Enrollment
   GROUP BY CourseID
) e ON c.CourseID = e.CourseID;
```

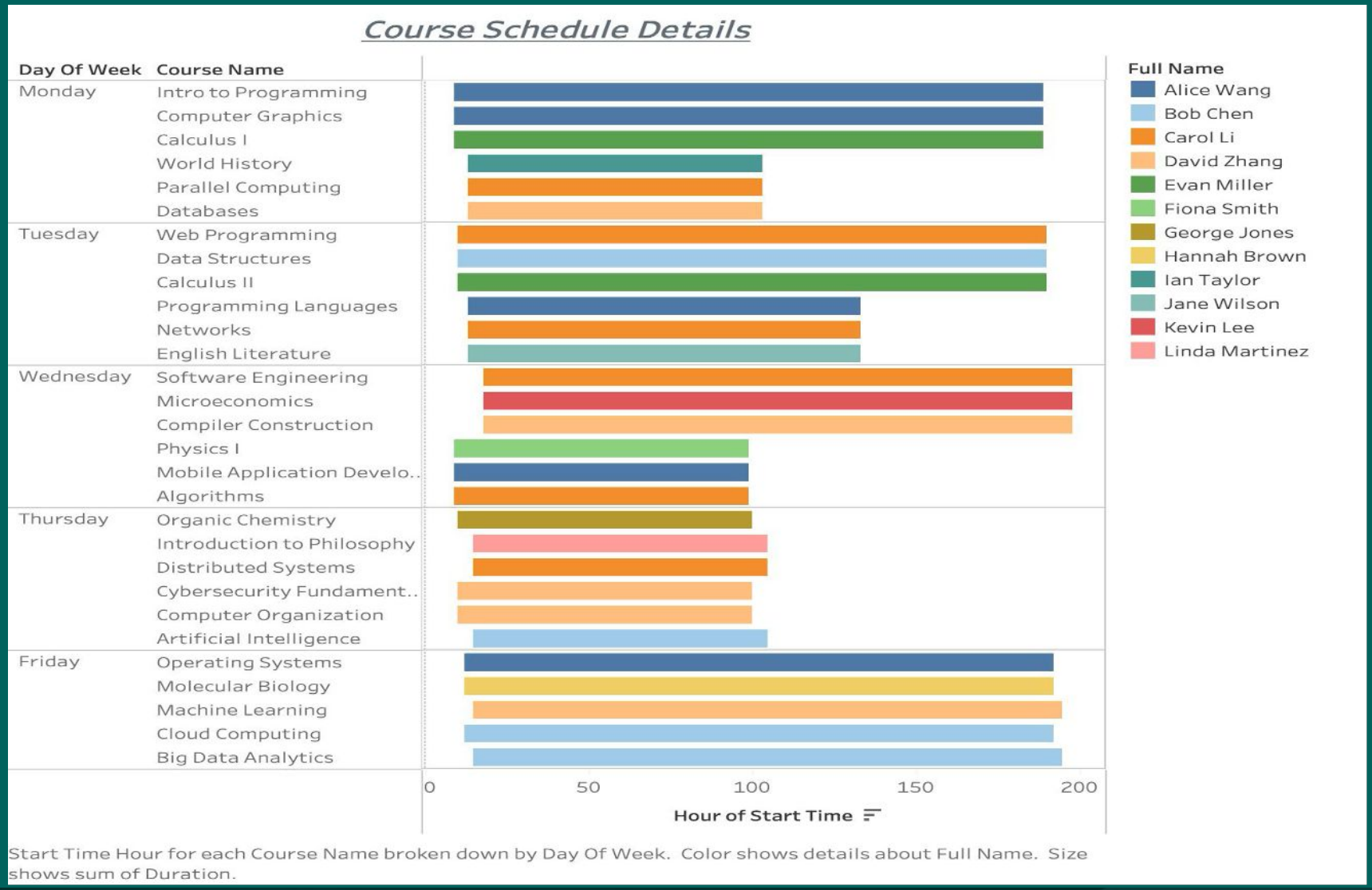
Visualization

Department and courses Details : *Packed Bubbles and Heatmap*



The packed bubbles and heatmap shows by the volume differentiation in the course enrollment number and the courses offered by the department.

Course Schedule Details: *Gantt Style Chart*



Course Schedule Details: Lists course schedules by day, time, instructor.

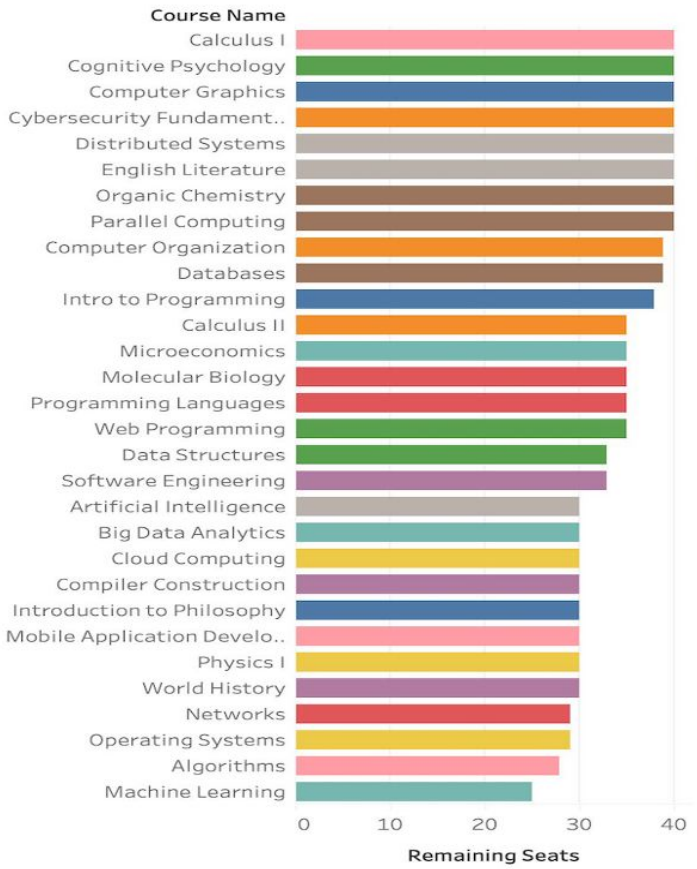
Hour of start time depicts the start date

The length of the bars depicts the duration(hours) of the course.

Color differentiation depicts the course is taught by which instructor.

Course Registration Details: *Horizontal Bars*

Remaining Seats by Course

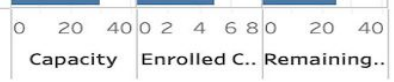


KPI View

Filled Rate %	2.56%
Total Capacity	1,055
Total Enrolled	27

Course Registration View

Regis..	Course Name	Semester Name
Closed	Parallel Computing	Summer 2024
	Organic Chemistry	Summer 2024
	Cybersecurity Fundamentals	Fall 2023
	Computer Graphics	Fall 2022
	Cognitive Psychology	Spring 2023
	Calculus I	Summer 2023
	Databases	Summer 2024
	Computer Organization	Fall 2023
	Intro to Programming	Fall 2022
	Web Programming	Spring 2023
	Programming Languages	Fall 2024
	Molecular Biology	Fall 2024
	Calculus II	Fall 2023
	Data Structures	Spring 2023
	Physics I	Spring 2024
	Mobile Application Development	Summer 2023
	Introduction to Philosophy	Fall 2022
	Cloud Computing	Spring 2024
	Operating Systems	Spring 2024
	Networks	Fall 2024
	Algorithms	Summer 2023
Open	English Literature	Summer 2025
	Distributed Systems	Summer 2025
	Microeconomics	Fall 2025
	Software Engineering	Spring 2025
	World History	Spring 2025
	Compiler Construction	Spring 2025
	Big Data Analytics	Fall 2025
	Artificial Intelligence	Summer 2025
	Machine Learning	Fall 2025

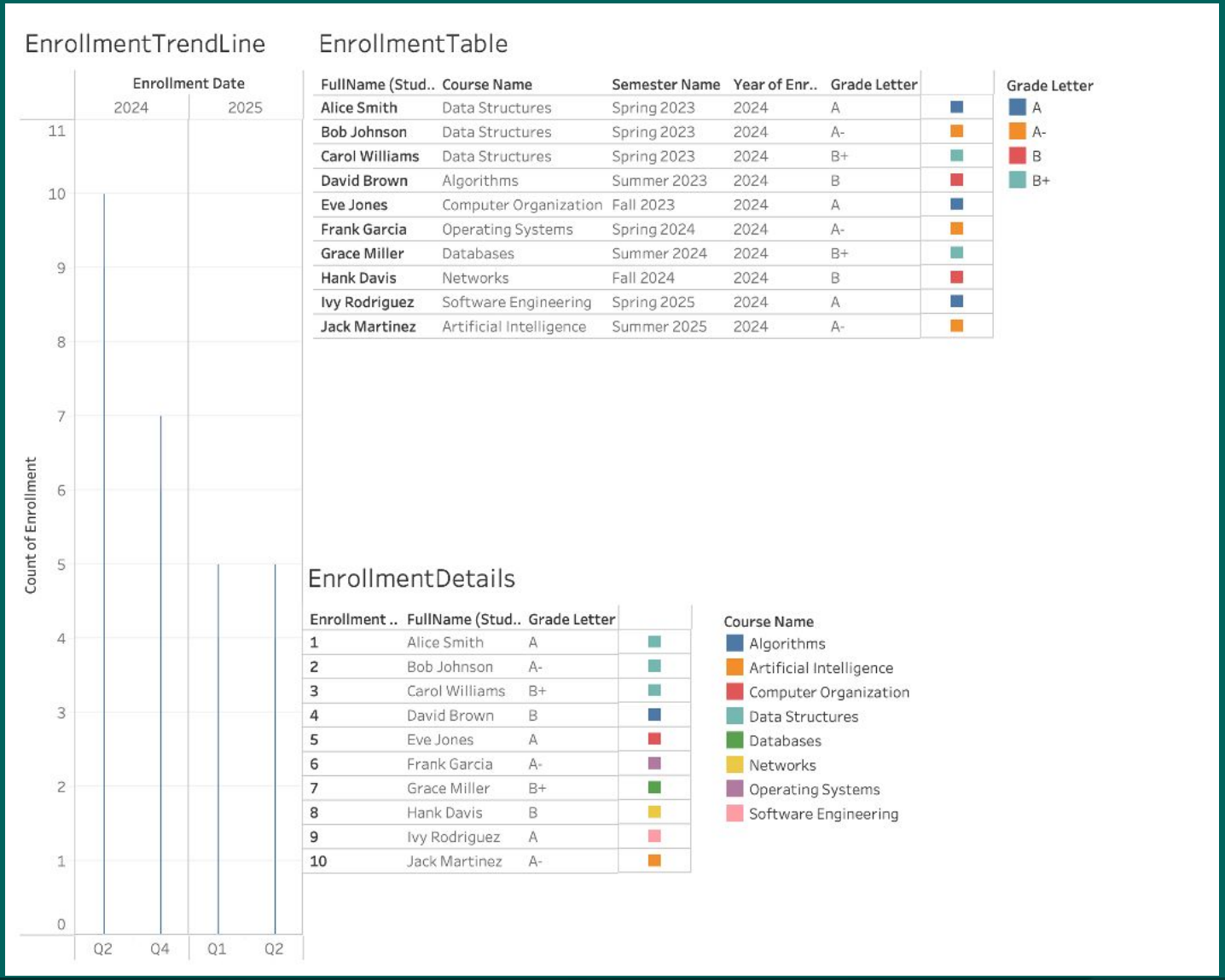


Semester Name
- Fall 2022
- Fall 2023
- Fall 2024
- Fall 2025
- Spring 2023
- Spring 2024
- Spring 2025
- Summer 2023
- Summer 2024
- Summer 2025

Registration Status
- Closed
- Open

Semester Name
- Fall 2022
- Fall 2023
- Fall 2024
- Fall 2025
- Spring 2023
- Spring 2024
- Spring 2025
- Summer 2023
- Summer 2024
- Summer 2025

Course Schedule Details: *Line Trend and Text Table*



Enrollment Details: Combines student, course, instructor, and grade information for enrollment history.

Color differentiation has been done for grades and Courses in the two tables.

The trend lines helps to understand total enrollment per quarter.

Wrap Up

Project Recap

Objective:

- Centralize and modernize course enrollment management.
- Streamline student registration, course management, and reporting.

Key Business Problems Addressed:

- Accurate student data management.
- Efficient course scheduling and enrollment.
- Prerequisite enforcement and waitlist processing.

Database Design Highlights

Core Entities:

- Student, Major, Instructor, Course, Enrollment, etc.

Visual Aid:

- Display simplified ERD or diagram.

Key Design Decisions:

- Use of Primary and Foreign Keys for integrity.
- Associative entities for many-to-many relationships (e.g., CoursePrerequisite, CourseSchedule).
- Adherence to Third Normal Form (3NF) to minimize data redundancy.

Implementation Details

SQL Code Overview:

- Database creation, table definitions, and data types.
- Insertion scripts with a minimum of 10 rows per table.

Views for Reporting:

- At least 2 views for summarizing enrollments and course details.

Advanced Features:

- Table-level CHECK Constraints using user-defined functions.
- Computed Columns (e.g., FullName combining FirstName and LastName).
- Data encryption for sensitive columns like passwords.

Impact & Benefits

Operational Benefits:

- Streamlined enrollment processes.
- Enhanced data integrity and security.
- Improved reporting capabilities for administration.

User Experience:

- Intuitive interfaces for students and instructors.
- Role-based access control for better security.

Future Enhancements & Learnings

Future Work:

- Further automation of reporting and analytics.
- Scalability improvements for larger institutions.
- Integration with additional modules (e.g., payment systems, alumni tracking).

Lessons Learned:

- Importance of normalization and integrity constraints.
- Benefits of advanced SQL features in real-world applications.

Call to Action:

- Open invitation for questions and discussion on improvements.

Thank You!!!

Q&A and Final Thoughts